LAPORAN PRAKTIKUM PEKAN 5

ALGORITMA DAN PEMROGRAMAN

PERULANGAN FOR (*FOR LOOPS*)

Disusun oleh:

Thaariq Salam

2511532022

Dosen Pengampu: Dr. Wahyudi S.T.M.T

Asisten Praktikum: Rahmad Dwirizki Olders

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga laporan praktikum mata kuliah Algoritma dan Pemrograman dengan topik "*For loops* (Perulangan *For*)" ini dapat disusun dan diselesaikan dengan baik. Laporan ini merupakan bagian dari pemenuhan tugas akademik pada Praktikum Pekan ke-5.

Topik *for loops* merupakan salah satu konsep dasar dalam pemrograman yang sangat penting untuk dipahami, karena memungkinkan eksekusi perintah secara berulang dengan kontrol yang efisien. Melalui praktikum ini, mahasiswa diharapkan mampu memahami struktur perulangan for, serta mengimplementasikannya dalam berbagai skenario pemrograman yang membutuhkan iterasi.

Penyusun menyadari bahwa tersusunnya laporan ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, ucapan terima kasih kami sampaikan kepada:

1. Bapak Dr. Wahyudi, S.T., M.T., selaku Dosen Pengampu mata kuliah Algoritma dan Pemrograman.
2. Saudara Rahmad Dwirizki Olders, selaku Asisten Praktikum, atas bimbingan dan arahannya selama pelaksanaan praktikum.

Penyusun berharap laporan ini tidak hanya menjadi bentuk pertanggungjawaban akademik, tetapi juga dapat menjadi referensi yang bermanfaat bagi pembaca dalam memahami konsep perulangan for. Kritik dan saran yang membangun sangat kami harapkan demi penyempurnaan laporan di masa mendatang.

Padang, 30 Oktober 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, kemampuan untuk mengulang suatu proses secara otomatis merupakan salah satu aspek fundamental yang sangat penting. Salah satu struktur kontrol yang digunakan untuk tujuan ini adalah *perulangan for* (*for loop*). Struktur ini memungkinkan eksekusi blok kode secara berulang dengan jumlah iterasi yang telah ditentukan, sehingga sangat berguna dalam berbagai skenario, seperti pemrosesan data dalam jumlah besar, pengulangan tugas-tugas rutin, hingga pembuatan pola dan simulasi[1].

Pemahaman yang baik terhadap konsep *for loop* tidak hanya membantu dalam menulis kode yang efisien dan terstruktur, tetapi juga menjadi dasar dalam mempelajari konsep pemrograman yang lebih kompleks seperti algoritma pencarian, pengurutan, dan manipulasi array[1]. Oleh karena itu, penguasaan materi ini menjadi sangat penting bagi mahasiswa dalam membangun fondasi logika pemrograman yang kuat.

Melalui praktikum ini, mahasiswa diharapkan dapat memahami sintaks dasar, logika kerja, serta penerapan *for loop* dalam berbagai kasus. Dengan latihan langsung melalui praktikum, mahasiswa dapat mengasah keterampilan pemrograman mereka dan meningkatkan kemampuan dalam menyusun solusi algoritma yang efektif dan efisien.

1.2 Tujuan

Praktikum pekan ke-5 ini bertujuan untuk memperkuat pemahaman mahasiswa terhadap konsep perulangan dalam pemrograman, khususnya struktur *for loop*. Adapun tujuan khusus dari praktikum ini adalah sebagai berikut:

1. Memahami prinsip dasar struktur perulangan *for* sebagai alat untuk mengotomatisasi proses berulang dalam program.
2. Mampu mengimplementasikan *for loop* dalam berbagai kasus pemrograman yang memerlukan iterasi dengan jumlah pengulangan tertentu.
3. Menguasai penggunaan *nested for loops* (perulangan bersarang) untuk menyelesaikan permasalahan yang melibatkan struktur data bertingkat atau pola berulang yang kompleks.
4. Meningkatkan kemampuan logika dan analisis dalam merancang solusi algoritma yang efisien dan adaptif terhadap variasi input data melalui penggunaan perulangan.

1.3 Manfaat

Praktikum pekan ke-5 mengenai *for loops* dan *nested loops* memberikan kontribusi yang signifikan dalam pengembangan kemampuan teknis dan logis mahasiswa. Manfaat utamanya adalah peningkatan dalam kemampuan berpikir terstruktur dimana mahasiswa dilatih untuk merancang solusi pemrograman yang efisien dalam mengolah data berulang.

Penguasaan atas struktur perulangan, seperti *for loop* dan *nested loop*, merupakan fondasi penting dalam membangun program yang mampu menangani proses berulang secara sistematis dan optimal. Melalui praktikum ini, mahasiswa tidak hanya belajar mengeksekusi perintah secara berulang, tetapi juga memahami bagaimana mengatur alur logika yang bersarang.

BAB II

PEMBAHASAN

2.1 Program Nested For0

2.1.1 Kode program

Bagian ini berisi kode sumber lengkap yang telah Anda implementasikan.

```
package Pekan5;
public class nestedFor0 {
public static void main(String[]args) {
        for (int line = 1; line<=5; line++) {
                for (int j = 1; j <= (-1 * line + 5); j++) {
                        System.out.print(".");
                        }
                System.out.print(line);
                System.out.println();
        }
}
}
```

2.1.2 Langkah kerja

Bagian ini menjelaskan secara terperinci tahapan eksekusi kode program dari awal hingga akhir, menitikberatkan pada mekanisme perulangan bersarang.

1) Inisialisasi perulangan luar (baris):
    - Perulangan for (int line = 1; line <= 5; line++) dimulai. Variabel line (baris) diinisialisasi dari 1 dan akan diulang selama line masih kurang dari atau sama dengan 5.
2) Eksekusi perulangan dalam (titik):

- Setiap kali perulangan luar berjalan, perulangan dalam for (int j = 1; j <= (-1 * line + 5); j++) dieksekusi.
- Perulangan ini berfungsi mencetak karakter titik (.).
- Batas perulangan dalam dihitung secara dinamis menggunakan ekspresi (-1 * line + 5).

3) Proses pencetakan:
   - Setelah perulangan dalam selesai (semua titik tercetak), program mencetak nilai dari variabel line (nomor baris saat ini).
   - Setelah mencetak nomor baris, system.out.println() dipanggil untuk pindah ke baris baru.
4) Pengulangan:
   - Nilai line ditingkatkan (line++), dan proses kembali ke langkah 1 hingga line mencapai nilai 6 (kondisi perulangan luar salah).

2.1.3 Analisis hasil

Program ini bertujuan untuk menghasilkan pola segitiga terbalik yang rata kanan dengan tinggi 5 baris. Pola ini dibentuk menggunakan karakter titik (.) sebagai spasi penyeimbang dan nomor baris sebagai karakter utama. Perulangan luar (line) mengontrol dimensi vertikal (tinggi/baris), sedangkan perulangan dalam (j) mengontrol dimensi horizontal (lebar/kolom) dari titik-titik pada setiap baris.

Ekspresi batas perulangan dalam: (-1 * line + 5) ini menciptakan hubungan berbanding terbalik antara nomor baris (line) dan jumlah karakter titik yang dicetak. Saat line bertambah (turun ke baris berikutnya), nilai ekspresi berkurang, yang berarti jumlah titik yang dicetak berkurang secara linear. Inilah yang menyebabkan pola titik membentuk segitiga terbalik, menempatkan nomor baris selalu di posisi paling kanan. Output program akan terlihat seperti berikut:

....1
...2
..3
.4
5

2.2 Program Nested For1

2.2.1 Kode program

Praktikum ini memperkenalkan konsep perulangan bersarang (*Nested Loops*), mengatasi keterbatasan perulangan tunggal dengan memungkinkan iterasi di dalam iterasi. Struktur kontrol ini esensial untuk tugas-tugas yang memerlukan pemrosesan data dua dimensi atau pembuatan pola terstruktur, menjamin setiap elemen dalam suatu dimensi (kolom) diproses secara penuh untuk setiap elemen pada dimensi lainnya (baris)[1].

```
package Pekan5;
public class nestedFor1 {
public static void main(String[]args) {
        for (int i=1 ; i<=5; i++) {
                for (int j=1; j<=5 ; j++) {
                        System.out.print("*");
                }
                System.out.println();
                // to end the line
        }
}
}
```

2.2.2 Langkah kerja

Kode ini bertugas mencetak pola persegi bintang (*) berukuran 5x5. Cara kerjanya sangat teratur:

1. Mengatur tinggi (loop luar i): Program memulai hitungan untuk Baris dengan variabel i dari 1 hingga 5. Ini menetapkan bahwa pola kita akan memiliki 5 baris.
2. Mengisi lebar (loop dalam j): Untuk setiap satu baris yang diatur oleh i, loop dalam yang diwakili oleh variabel j akan berjalan dari 1 hingga 5.Loop j ini

bertugas mencetak 5 bintang (*) secara horizontal (ke samping) di baris tersebut.

3. Pindah ke baris baru: Setelah 5 bintang (*) selesai dicetak (loop j selesai), perintah System.out.println(); akan dipanggil. Ini seperti tombol Enter yang membuat kursor pindah ke awal baris berikutnya.
4. Ulangi proses: Proses ini berulang sebanyak 5 kali (sesuai loop i), menghasilkan 5 baris bintang yang sama panjang.

2.2.3 Analisis hasil

Loop luar (i) dan loop dalam (j) sama-sama memiliki batas hitungan yang tetap (sampai 5) karena batas perulangan dalam (j <= 5) tidak pernah berubah meskipun nilai i (baris) bertambah, setiap baris selalu dicetak dengan jumlah karakter yang sama (5 bintang). Ini menghasilkan bentuk persegi 5x5. Total bintang yang dicetak adalah 5x5 = 25 bintang. Output program akan terlihat seperti berikut:

```
*****
*****
*****
*****
*****
```

2.3 Program Nested For2

2.3.1 Kode program

Program ini melanjutkan Perulangan Bersarang (Nested Loops), namun kali ini dengan kondisi perulangan dalam yang berbeda. Hal ini memungkinkan program untuk membuat pola yang tidak seragam (misalnya, segitiga) di mana jumlah kolom bergantung pada nomor baris saat ini.

```
package Pekan5;
public class nestedFor2 {
public static void main(String[]args) {
        for (int i=0 ; i<=5; i++) {
```

```
                for (int j=0; j<=5 ; j++) {
                        System.out.print(i+j+ " ");
                }
                System.out.println();
        }
}
}
```

2.3.2 Langkah kerja

Kode ini membuat tabel hasil penjumlahan dengan dimensi 6x6 (karena indeks dimulai dari 0 hingga 5).

1. Mengatur baris (loop luar i):
    - Loop luar dengan variabel i diinisialisasi dari 0 hingga 5 (total 6 iterasi). Ini menentukan baris tabel.
2. Mengatur kolom (loop dalam j):
    - Untuk setiap baris, loop dalam dengan variabel j juga berjalan dari 0 hingga 5 (total 6 iterasi). Ini menentukan kolom tabel.
3. Proses pencetakan:
    - Di dalam loop terdalam, program mencetak hasil dari i + j, diikuti dengan spasi (" ").
4. Pindah baris:
    - Setelah loop dalam (j) selesai mencetak 6 nilai, system.out.println(); dipanggil untuk pindah ke baris baru.
5. Ulangi proses:
    - Proses ini diulang hingga loop luar (i) selesai mencapai nilai 5.

2.3.3 Analisis hasil

Program ini menghasilkan tabel penjumlahan yang berisi 6x6 angka. Nilai pada setiap sel adalah hasil penjumlahan indeks baris (i) dan indeks kolom (j). Karena

batas kedua loop (luar dan dalam) adalah tetap (sama-sama 6 iterasi), output yang dihasilkan adalah pola persegi (6x6).

- Nilai yang dicetak adalah hasil i + j:
    - Nilai terkecil adalah 0+0 = 0 (baris 0, kolom 0).
    - Nilai terbesar adalah 5+5 = 10(baris 5, kolom 5).
    - Tabel ini menunjukkan relasi antara dua dimensi: nilai di setiap baris bertambah secara linear, dan nilai di setiap kolom juga bertambah secara linear.
- Kode ini membuktikan bahwa perulangan bersarang tidak hanya untuk pola bintang, tetapi juga efektif untuk memproses data matriks atau melakukan operasi yang melibatkan dua indeks secara bersamaan, membentuk struktur data dua dimensi. Output program akan terlihat seperti berikut:

0 1 2 3 4 5
1 2 3 4 5 6
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10

2.4 Program Perulangan For1

Program ini berfungsi sebagai demonstrasi paling dasar dari perulangan for tunggal. Tujuannya adalah memastikan pemahaman Anda tentang tiga komponen kunci dalam struktur for untuk menjalankan kode secara berulang dengan hitungan yang jelas.

2.4.1 Kode program

```
package Pekan5;
public class PerulanganFor1 {
```

```
public static void main(String[]args) {
        for (int i=1;i<=10;i++) {
                System.out.println(i);
        }
}}
```

2.4.2 Langkah kerja

Kode ini akan mencetak angka dari 1 hingga 10, masing-masing pada baris baru.

1) Inisialisasi (int i=1): hitungan dimulai dari variabel i yang nilainya diatur ke 1.
2) Kondisi (i<=10): selama nilai i masih kurang dari atau sama dengan 10, perulangan akan terus berjalan.
3) Aksi (system.out.println(i)): angka saat ini (i) dicetak, diikuti dengan pindah baris (println).
4) Increment (i++): setelah aksi selesai, nilai i ditingkatkan 1.
5) Pengulangan: langkah 2, 3, dan 4 diulang hingga kondisi (langkah 2) menjadi salah (yaitu, ketika i menjadi 11).

2.4.3 Analisis hasil

Program ini menunjukkan inti dari struktur perulangan for: efisiensi dalam mengotomatisasi tugas yang berulang. Mencetak urutan angka 1 sampai 10, hanya dengan 3 baris kode (di dalam *main*), program berhasil menjalankan perintah cetak sebanyak 10 kali tanpa menulis 10 baris system.out.println().

Keberhasilan perulangan ini terletak pada kondisi batas (i<=10) yang terdefinisi dengan jelas, yang dikombinasikan dengan mekanisme *increment* (i++) untuk memastikan perulangan tersebut berakhir dan tidak menjadi *infinite loop* (perulangan tak terbatas). Output program akan terlihat seperti berikut:

1
2
3
4
5
6
7
8
9
10

2.5 Program Perulangan For2

Program ini berfungsi sebagai demonstrasi paling dasar dari perulangan for tunggal.

2.5.1 Kode program

```
package Pekan5;
public class PerulanganFor2 {
public static void main(String[]args) {
        for (int i=1;i<=10;i++) {
                System.out.print(i+" ");
        }
}
}
```

2.5.2 Langkah kerja

Kode ini akan mencetak angka dari 1 hingga 10 dalam satu baris, dipisahkan oleh spasi.

1. Inisialisasi (int i=1): hitungan perulangan dimulai dari variabel i dengan nilai awal 1.

2. Kondisi (i<=10): perulangan terus berjalan selama nilai i masih kurang dari atau sama dengan 10.
3. Aksi (system.out.print(i+" ")): angka saat ini (i) dicetak, diikuti dengan satu spasi (" "). Penggunaan fungsi print() (bukan println()) adalah kunci agar output tetap dalam satu baris.
4. Increment (i++): setelah aksi selesai, nilai i ditingkatkan 1.
5. Pengulangan: proses ini diulang hingga kondisi (langkah 2) menjadi salah (yaitu, ketika i menjadi 11).

2.5.3 Analisis hasil

Program ini menunjukkan inti dari struktur perulangan for dalam mengotomatisasi tugas berulang dan bagaimana kontrol output memengaruhi tata letak hasil. Mencetak urutan angka 1 sampai 10 secara horizontal. Perbedaan utama dengan println() adalah penggunaan System.out.print(). Ini memungkinkan seluruh output tetap dalam satu aliran horizontal, dengan spasi (" ") berfungsi sebagai pemisah antar angka.

Perulangan ini berhasil menjalankan aksi cetak sebanyak 10 kali hanya dengan satu baris kode, menunjukkan efisiensi dalam penulisan program. Output program akan terlihat seperti berikut:

1 2 3 4 5 6 7 8 9 10

2.6 Program Perulangan For3

Program ini merupakan demonstrasi perulangan for yang digunakan untuk menghitung dan menampilkan jumlah deret bilangan dari 1 hingga 10. Program ini juga menunjukkan penggunaan kondisi if sederhana di dalam perulangan untuk mengatur tata letak output.

2.6.1 Kode program

```
package Pekan5;
public class PerulanganFor3 {
	public static void main(String[]args) {
		int jumlah=0;
		for (int i=1;i<=10;i++) {
			System.out.print(i);
			jumlah= jumlah+i;
			if (i<10) {
				System.out.print(" + ");
			}
		}
		System.out.println();
		System.out.println("Jumlah = " +jumlah);
	}
}
```

2.6.2 Langkah kerja

Kode ini menghitung total penjumlahan angka 1 hingga 10 sambil mencetak deretnya dalam format penjumlahan.

1. Inisialisasi variabel: variabel jumlah diinisialisasi ke 0. Variabel ini berfungsi sebagai akumulator untuk menyimpan hasil total penjumlahan.
2. Perulangan (for loop): perulangan dimulai dari i=1 hingga i=10.
3. Cetak angka: di setiap iterasi, nilai i dicetak (system.out.print(i)).
4. Akumulasi: nilai i saat ini ditambahkan ke variabel jumlah (jumlah = jumlah + i).
5. Cetak tanda tambah: program menggunakan pernyataan if (i < 10) untuk mengontrol format output. Jika i belum mencapai 10, tanda " + " dicetak. Hal ini memastikan tanda tambah tidak dicetak setelah angka terakhir (10).

6. Cetak hasil akhir: setelah perulangan selesai, program pindah baris (system.out.println();) dan mencetak nilai akhir dari variabel jumlah.

2.6.3 Analisis hasil

Program ini menunjukkan fungsionalitas ganda dari perulangan: iterasi dan komputasi. Variabel jumlah berfungsi sebagai akumulator. Dalam setiap putaran, ia mengambil nilai sebelumnya dan menambahkan nilai i yang baru, secara efektif menghitung total deret (1 + 2 + 3 + ... + 10). Penggunaan pernyataan *if* di dalam for loop adalah kunci untuk menghasilkan output yang rapi. Kondisi if (i < 10) adalah teknik yang efisien untuk mencegah pencetakan karakter pemisah (+) pada elemen terakhir deret, sehingga output tampak seperti persamaan matematika yang benar. Hasil akhir menunjukkan deret yang dicetak dan total penjumlahan yang dihitung. Output program akan terlihat seperti berikut:

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
Jumlah = 55
```

2.7 Program Perulangan For4

Program ini adalah contoh perulangan for yang digunakan untuk menghitung jumlah deret bilangan dari 1 hingga batas yang ditentukan oleh input pengguna. Program ini menunjukkan cara mengintegrasikan interaksi pengguna (*user input*) dengan struktur perulangan.

2.7.1 Kode program

```
package Pekan5;
import java.util.Scanner;
public class PerulanganFor4 {
        public static void main(String[]args) {
                int jumlah=0;
```

```
                int batas;
                Scanner input =new Scanner (System.in);
                System.out.print("Masukkan nilai batas = ");
                batas= input.nextInt();
                input.close();
                for (int i=1;i<=batas;i++) {
                        System.out.print(i);
                        jumlah= jumlah + i;
                        if (i<batas) {
                                System.out.print(" + ");
                        }else {
                                System.out.print(" = ");
                        }
                }
         System.out.print(jumlah);
        }
}
```

2.7.2 Langkah kerja

Kode ini menghitung total penjumlahan angka dari 1 hingga batas tertentu yang dimasukkan oleh pengguna, sambil menampilkan deretnya.

1. Inisialisasi & input: program mendeklarasikan variabel jumlah (akumulator) dan batas. Kemudian, ia meminta pengguna memasukkan nilai batas menggunakan objek scanner.
2. Perulangan dinamis (for loop): perulangan dimulai dari i=1 dan akan terus berjalan selama i masih kurang dari atau sama dengan nilai batas yang dimasukkan pengguna.
3. Cetak & akumulasi: di setiap iterasi, nilai i dicetak dan ditambahkan ke variabel jumlah.
4. Kontrol tata letak:

- Pernyataan if (i < batas) mencetak tanda + jika i belum mencapai angka terakhir.
- Pernyataan else dijalankan hanya ketika i sudah mencapai batas, mencetak tanda = untuk menutup deret penjumlahan.

5. Cetak hasil akhir: setelah perulangan selesai, nilai akhir dari jumlah dicetak, menyelesaikan persamaan.

2.7.3 Analisis hasil

Program ini menunjukkan implementasi perulangan yang adaptif (*flexible*) dan integrasi struktur kontrol (*if-else*) di dalamnya. Kunci analisis di sini adalah bahwa kondisi perulangan i <= batas bersifat dinamis, artinya jumlah perulangan tidak tetap x kali, melainkan tergantung pada nilai yang diinput pengguna. Ini membuat program jauh lebih fleksibel. Pernyataan if-else digunakan di dalam *loop* untuk tujuan pemformatan output. Hal ini membuktikan bahwa dua struktur kontrol ini dapat bekerja sama untuk memecahkan masalah.

Jika pengguna memasukkan 5, output program akan terlihat seperti berikut:

Masukkan nilai batas = 5
1 + 2 + 3 + 4 + 5 = 15

BAB III

PENUTUP

3.1 Kesimpulan

Praktikum pekan 5 mengenai perulangan (*looping*) dan perulangan bersarang (*nested loops*) telah memperkuat pemahaman mendalam tentang cara kerja kontrol alur program untuk tugas yang berulang. Dari implementasi dan analisis yang telah dilakukan, dapat ditarik beberapa poin kunci:

1. Esensi perulangan for: struktur for adalah alat fundamental untuk mengotomatisasi eksekusi blok kode dengan jumlah iterasi yang telah ditentukan (1). Struktur ini sangat efisien dalam mencetak deret atau melakukan komputasi akumulatif (penjumlahan) hingga batas tertentu.
2. Perulangan bersarang (*nested loops*): *nested loops* merupakan solusi penting untuk menangani permasalahan yang melibatkan dimensi ganda (vertikal dan horizontal), seperti pembuatan pola (persegi, segitiga) atau pemrosesan data matriks (1). Hasil pola bergantung pada apakah kondisi *loop* dalam bersifat tetap (menghasilkan persegi) atau dinamis (menghasilkan segitiga).
3. Integrasi kontrol: perulangan dapat bekerja efektif dengan struktur kontrol lain (if-else). Penggunaan if di dalam for loop, seperti pada program deret penjumlahan dinamis, memungkinkan pengendalian output yang detail dan adaptif terhadap input pengguna.
4. Kontrol alur efisien: penguasaan struktur kontrol dasar, baik perulangan maupun pernyataan kondisi (if-else) adalah krusial. Struktur ini memungkinkan perancangan algoritma yang efisien (melalui mekanisme *short-circuiting*) dan adaptif terhadap berbagai skenario input (1).

Secara keseluruhan, kompetensi dalam mengelola perulangan dan mengintegrasikannya dengan pernyataan kondisi membekali mahasiswa untuk membangun fondasi logika pemrograman yang kuat dan merancang solusi yang sistematis dan optimal.

DAFTAR PUSTAKA

[1] P. Deitel dan H. Deitel, Java How to Program, Early Objects, 11th ed. Boston, MA: Pearson, 2018